Janne Nissinen

SUOSITTELUJÄRJESTELMÄN OHJELMOINTI

SUOSITTELUJÄRJESTELMÄN OHJELMOINTI

Janne Nissinen
Opinnäytetyö
Kevät 2014
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Janne Nissinen
Opinnäytetyön nimi: Suosittelujärjestelmän ohjelmointi
Työn ohjaaja: Markku Rahikainen
Työn valmistumislukukausi ja -vuosi: Kevät 2014
Sivumäärä: 29 + 2 liitettä

Opinnäytetyönä tuli toteuttaa suosittelujärjestelmä SingOn Oy:n nettikaraoke-palveluun. Suosittelujärjestelmän avulla pystytään keräämään tietoa käyttäjien laulamista kappaleista ja kohdistamaan ennalta määritellyn musiikkityylin tai käyttäjäluokan pohjalta sopivia vaihtoehtoja käyttäjälle laulettavaksi.

Suunnitelmana oli hyödyntää valmista suosittelujärjestelmää tai vaihtoehtoisesti ohjelmoida sellainen itse. Lisäksi suosittelujärjestelmä tuli integroida SingOn-karaokepalvelun serverille, josta sitä pystyttiin ajamaan ennalta määrättyinä hetkinä.

Opinnäytetyö aloitettiin perusteellisella tutkimuksella, jossa selvitettiin jo olemassa olevien suosittelujärjestelmien soveltuvuus yrityksen käyttötarkoituksiin. Vastaavasti oman suosittelujärjestelmän ohjelmoimiseen tarvittavat resurssit ja vaatimukset kirjattiin ylös. Lopuksi tehtiin yhteenveto ja suosittelujärjestelmäksi valittiin Myrrix-suosittelujärjestelmä, jonka tutkimisesta ja käyttöönotosta opinnäytetyö pääpiirteittäin muodostui.

Asiasanat**:** tietokannat, JavaScript, Ajax, REST

SISÄLLYS

SANASTO

AJAX	Lyhenne sanoista Asynchronous Javascript and XML. Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.
Back End	Taustalla toimiva ohjelmiston osa, jonka kanssa käyttäjä ei ole suoraan tekemisissä, taustaosa.
Big Data	Suuri (aloitus)datatietue, jossa datapisteiden määrä > 100 milj.
Cold Start	Termi, jolla kuvataan suosittelujärjestelmien ensikäynnistyksen yhteydessä esiintyvää ongelmaa, jossa kerättyä dataa ei ole riittävästi suosittelujen laskemiseksi.
Collaborative filtering	Suosittelujärjestelmien datan keräämislogiikka, jossa käyttäjät itse toimivat datan tuottajina omilla toiminnoillaan.
CSV	Tiedosto, jossa eri arvot erotetaan pilkulla. Käytetään mm. Excel-ohjelmassa.
Data Point	Käytetään kuvaamaan käyttäjien ja tuotteiden välisten yhteyksien määrää.
Front end	Ohjelmiston puoli, joka on näkyvissä käyttäjälle, ja jonka kanssa käyttäjä on vuorovaikutuksessa, käyttäjäosa.
HTTP	Hypertext Transfer Protocol. Protokolla,

	jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
Hybrid Recommender	Hybridijärjestelmä, useiden erilaisten suosittelujärjestelmien muodostama kokonaisuus
Initial dataset	Aloitusdata, jonka avulla suosittelujärjestelmät alkavat laskea suositteluja käyttäjille.
JavaScript, JS	Web-ympäristöissä käytettävä kieli, mahdollistaa sivustojen dynaamisen toiminnollisuuden.
JSON	Lyhenne sanoista JavaScript Object Notation. Yksinkertainen tiedonsiirtomuoto, jota käytetään JavaScript-ohjelmissa
Lähin naapuri	Tekniikka, jonka perusteella löydetään käyttäjän toimia ja mielenkiintoja yhdenmukaistava toinen (tai n:s) käyttäjä
PHP	PHP: Hypertext Preprocessor on Perlin kaltainen ohjelmointikieli, jolla pystytään tuottamaan dynaamista sisältöä websivustoille.
REDIS	Avoimeen lähdekoodiin perustuva muistissa ajettava tietokanta.
REST	Representational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli, jolla toteutetaan ohjelmointirajapintoja.

1 JOHDANTO

Suoritin opinnäytetyöni SingOn Oy:ssä, jossa suoritin aikaisemmin yrityslähtöiset harjoitteluni. Opinnäytetyön tekeminen yritykseen oli looginen jatkumo, sillä työympäristön ja laitteistojen ennalta tunteminen helpotti varsinaiseen kirjoitustyöhön keskittymistä. Lisäksi ehdotettu aihe oli mielenkiintoinen, mikä vaikutti valintaan ratkaisevasti.

Opinnäytetyön kantavana ajatuksena oli toteuttaa SingOn-verkkokaraokepalveluun suosittelujärjestelmä, joka antaa käyttäjille ehdotuksia kappaleista. Ehdotukset perustuvat käyttäjän laulamiin kappaleisiin, ja näiden perusteella tehdään älykkäitä arvauksia niistä kappaleista, joista käyttäjä voisi myös pitää.

Suosittelujärjestelmä antaa yritykselle tietoa käyttäjistä, ja tätä tietoa voidaan hyödyntää takaisin käyttäjiin. Esimerkiksi käyttäjien mieltymyksiin ja intresseihin perustuvaa sisältöä voidaan näyttää tehokkaammin, kun seurataan käyttäjien tarkastelemia, ostamia ja arvostelemia tuotteita. Esimerkkinä toimivat monet verkkokaupat, joissa suosittelujärjestelmä näyttää käyttäjille erilaisia vaihtoehtoja käyttäjän omien valintojen ja selaamien tuotteiden perusteella. Vaihtoehtoisesti voidaan myös ryhmitellä käyttäjiä ja ehdottaa tuotevaihtoehtoja muiden samanlaisten ryhmänjäsenten selaamien tuotteiden perusteella.

SingOn-karaokepalvelussa suosittelujärjestelmä toimii täysin kappaleiden ja artistien pohjalta ja antaa käyttäjän tai käyttäjäryhmien tekemien valintojen perusteella älykkäitä arvauksia eli ehdotuksia käyttäjilleen. Tavoitteena on kerätä tietoa käyttäjien laulamista kappaleista ja syöttää ne suosittelujärjestelmään. Tällöin suosittelujärjestelmä osaa muodostaa erilaisia käyttäjäryhmiä, joissa samanlaiset käyttäjät ja heidän kappaleensa sijaitsevat. Näin pystytään suosittelemaan käyttäjälle X laulettavaksi kappaleita, joita käyttäjä Y samassa ryhmässä on laulanut.

Opinnäytetyön rajojen määrittelemiseen käytettiin apuna suosittelujärjestelmän toiminnallisuutta. Suosittelujärjestelmän tuli toimia SingOn-palvelussa sisäisenä toimintona, ja sitä piti pystyä kutsumaan jo valmiina olevien funktioiden pohjalta. Suosittelujärjestelmän tuli antaa käyttäjälleen ehdotuksia, ja suosittelujärjestelmää tuli pystyä päivittämään mieluiten reaaliaikaisesti. Suosittelujärjestelmän tuli olla testattu ja tuotannossa käytössä opinnäytetyön päättyessä. Suosittelujärjestelmän mahdolliset laajennukset jätettiin tulevaisuuden varaan.

Opinnäytetyö aloitettiin perusteellisella tutkimuksella, jossa selvitettiin jo olemassa olevien suosittelujärjestelmien soveltuvuus yrityksen käyttötarkoituksiin. Vastaavasti oman suosittelujärjestelmän ohjelmoimiseen tarvittavat resurssit ja vaatimukset kirjattiin ylös. Lopuksi tehtiin yhteenveto ja valittiin yrityksen ja opinnäytetyöhön varatun ajan suhteen optimaalisin ratkaisu. Opinnäytetyö keskittyy valitun suosittelujärjestelmän käyttöönottoon ja testaamiseen.

2 SUOSITTELUJÄRJESTELMÄT

Suosittelujärjestelmien toimintaperiaatteet voidaan jakaa kahteen erilaiseen toimintatapaan: sisältöperustaiseen sekä yhteistoiminnalliseen suodattamiseen. On olemassa myös hybridijärjestelmiä, jotka yhdistävät kaksi edellä mainittua järjestelmää yhteen toimivaan kokonaisuuteen. Hybridijärjestelmissä pystytään varautumaan suosittelujärjestelmien haastavimpaan ongelmaan: yhteistoiminnallisten järjestelmien ensikäynnistykseen. (1.)

2.1 Sisältöpohjainen suosittelujärjestelmä

Sisältöperustaisissa toimintatavoissa lähtökohtana on käyttäjien ja tuotteiden profilointi, jonka perusteella voidaan tehdä automaattisia ennusteita käyttäjän kiinnostuksen kohteista. Vaikka sisältöperustaisen suosittelujärjestelmän toteuttaminen on helpompaa kuin yhteistoiminnallisen, on jälkimmäisen jatkokehittäminen mahdollista erilaisilla algoritmeilla. Lisäksi sisältöperustaisen suosittelujärjestelmän tehokkuus heikkenee kuluttajien ja tuotteiden määrän kasvaessa. (2, s. 310.)

2.2 Yhteistoiminnallinen suosittelujärjestelmä

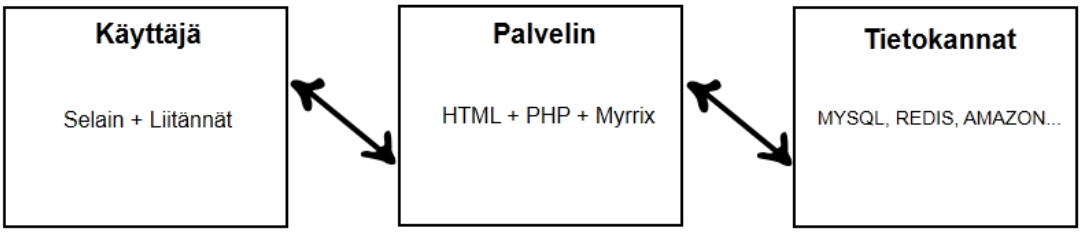
Yhteistoiminnallisissa järjestelmissä käytetään erilaisia matemaattisia algoritmeja, joiden perusteella voidaan ennustaa käyttäjän kiinnostuksen kohteita. Yhtenä esimerkkinä toimii Lähin naapuri -tekniikka, jossa etsitään käyttäjän vertaisia toisia käyttäjiä, joilla on aiemmin ollut taipumuksena olla samaa mieltä käyttäjän kanssa. Käytännössä naapurit muodostuvat käyttäjän mielipiteiden keskinäisestä korrelaatiosta. Tämän jälkeen järjestelmä osaa suositella käyttäjälle tuotteita naapureiden tekemien valintojen perusteella. (2, s. 319.)

2.3 Suosittelujärjestelmän toimintaperiaate

Jotta tulevan suosittelujärjestelmän implementointi onnistuisi mahdollisimman tehokkaasti, täytyi suunnitteluvaiheessa ottaa huomioon järjestelmien toimintaperiaatteet. Yleisesti voidaan ajatella, että suosittelujärjestelmien toimintamal-

leissa ei ole suuria poikkeamia, joten tarkastelussa ei ole erikseen huomioitu jokaisen mahdollisen järjestelmän mallia.

Suosittelujärjestelmä toimii taustaosapalveluna, jota kohdesivuston erilliset funktiot kutsuvat tarvittaessa. Kun käyttäjälle halutaan tuottaa suositteluja, lähetetään palveluun pyyntö, johon suosittelujärjestelmä vastaa suositteluilla. Vastauksen jälkiprosessointi tapahtuu myös palvelun taustaosassa, ja lopputulos näytetään halutussa muodossa käyttäjän näkemälle käyttäjäosalle. (Kuva 1.)



KUVA 1. Palvelimen arkkitehtuurimalli

Jotta suosittelujärjestelmä kykenee laskemaan suositteluja käyttäjälle, se tarvitsee aloitusdataa. Aloitusdatana yleensä käytetään jo valmiiksi kerättyä dataa käyttäjien tapahtumista sivulla, kuten esimerkiksi klikkauksista, ostotapahtumista, arvosteluista ja tarkasteluista. Suosittelujärjestelmiä voidaan käyttää myös ilman aloitusdataa, mutta tällöin vastaan tulee useita ongelmia, kuten mm. ensikäynnistysongelma, jolloin suosittelujärjestelmässä ei ole tarpeeksi dataa, jotta suositteluja pystyttäisiin luotettavasti laskemaan. Opinnäytetyössä käytetyn aloitusdatan muotoilu näkyy CSV-tiedostosta otetusta kuvasta (ks. kuva 2), jossa luvut vastaavat käyttäjä- ja kappaletunnusta.

A
3400,2266
3399,2429
3399,2429
3402,2426
3402,226
3402,2432
3402,2432
3398,2258
3398,2317
3838,2387
3838,2429
3838,2297

KUVA 2. CSV-tiedoston kuvakaappaus

Suosittelujärjestelmien dataliikenne toimii kahteen suuntaan. Dataa voidaan saada ulos suosittelujen muodossa, mutta dataa voidaan myös syöttää sisään reaaliaikaisesti, jolloin suosittelujärjestelmä oppii uusia liitoksia käyttäjien ja tuotteiden välillä, eli datapisteitä, ja pystyy päivittämään tietokantansa.

Jälkimmäinen toiminto on tärkeä, mikäli kohdesivuston tuotetarjonta kehittyy jatkuvasti. Mikäli toimintoa ei pystyttäisi suorittamaan, suosittelujärjestelmä jättäisi uudet tuotteet kokonaan pois suositeltavista tuotteista eivätkä suositeltavat kappaleet koskaan päivittyisi.

2.4 Suosittelujärjestelmän valitseminen

Koska SingOn-karaokepalveluun rekisteröityy jatkuvasti uusia käyttäjiä ja lauletta-via kappaleita lisätään säännöllisesti, ei pelkkää sisältöperustaista suosittelu-järjestelmää voida hyödyntää tehokkaasti palvelussa. Tämän johtopäätöksen seurauksesta kaikki pelkkää sisältöperustaista toimintamallia käyttävät järjes-telmät jätettiin huomioimatta.

Yhteistoiminnalliset järjestelmät vaikuttivat rationaalisimmilta vaihtoehdoilta, mutta niiden ongelmat täytyi huomioida. Järjestelmissä keskeisimmäksi ongel-maksi muodostui niiden ensikäynnistysongelma. Ongelmassa järjestelmän käyt-töönottovaiheessa ei ole tarpeeksi dataa, jonka perusteella voitaisiin antaa te-

hokkaita vastauksia. Kuitenkin järjestelmien toimivuus laajemman datan kanssa, erilaisten algoritmien kanssa yhteensopivuus ja laajennettavuus olivat kriteerejä, jotka auttoivat päättämisessä. Suosittelujärjestelmäksi valittiin Myrrix-suosittelujärjestelmä, joka on jatkokehitetty Apachen valmistamista Hadoop- ja Mahout-järjestelmistä.

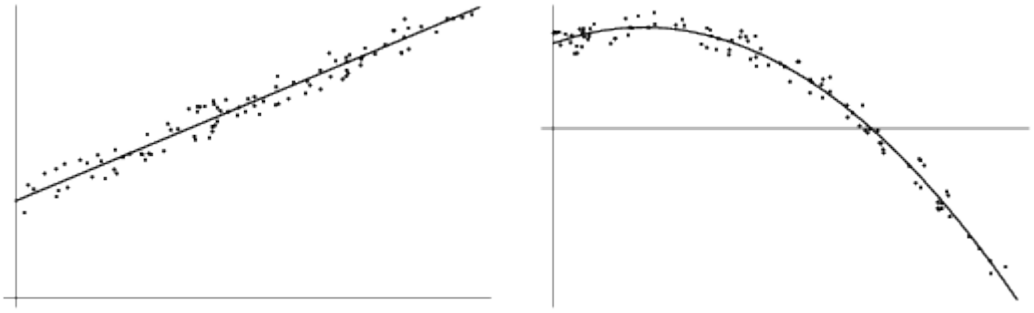
3 MYRRIX-SUOSITTELUJÄRJESTELMÄ

3.1 Myrrix-suosittelujärjestelmän rakenne

Myrrix-suosittelujärjestelmä on eräänlainen hybridijärjestelmä, joka pohjautuu Apache-yrityksen luomiin Hadoop- sekä Mahout-järjestelmiin. Myrrix on kirjoitettu Javalla, ja siihen on saatavilla REST-pohjaisia rajapintoja eri kielille. Myrrix-järjestelmä toimii kahdessa tasossa, jotka on nimetty laskenta- ja palvelutasoksi.

3.1.1 Laskentataso

Laskentataso on taustalla toimiva kokonaisuus, jossa muun muassa lasketaan käyttäjien ja tuotteiden välisistä datapisteistä lineaarisella regressioanalyysillä jana. Janan laskemisessa optimaalisin lopputulos on tilanne, jossa jäännösvirheen neliöiden summa on minimaalisin. (Kuva 3.)



KUVA 3. Esimerkkikuva lineaarisesta regressioanalyysistä 50 datapisteelle (4.)

Laskentataso toimii myös eräänlaisena tietokantana, jossa käyttäjien ja tuotteiden väliset datapisteet säilötään. Tietokantaa voidaan päivittää syöttämällä suosittelujärjestelmälle datapisteitä palvelutason kautta. (5.)

3.1.2 Palvelutaso

Palvelutaso toimii välikätenä pyynnöille ja toimittaa ne eteenpäin laskentatasolle. Palvelutasoa voidaankin kutsua eräänlaiseksi rajapinnaksi laskentatason funktioihin, jossa suosittelut lasketaan. Palvelutaso pystyy palvelemaan useaa

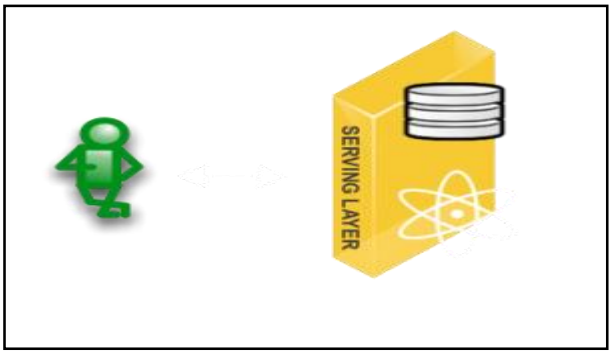
käyttäjää samanaikaisesti ja välittää niiden pyynnöt yhdelle tai useammalle laskentatasolle.

3.2 Myrrix-suosittelujärjestelmän käyttö

Myrrix-suosittelujärjestelmän tasot on luotu siten että niitä voidaan ajaa yhdessä tai pelkkää palvelutasoa voidaan ajaa itsenäisesti. Tasojen yhdessä ajamista kutsutaan yhdistetyksi tilaksi, kun pelkän palvelutason ajamista kutsutaan itsenäiseksi tilaksi.

3.2.1 Itsenäinen tila

Itsenäisessä tilassa palvelutaso toimii itse myös laskentatasona ja suorittaa kaikki matemaattiset algoritmit itsenäisesti. Palvelutaso toimii myös tietokantana ja säilyttää kaikki datapisteet itsensä sisällä (ks. kuva 4). Tila on kevyempi versio yhdistetystä tilasta ja sitä on helpompi käyttää ja hallita, mutta sen luomat rajoitteet käyttäjien ja datapisteiden määrissä voivat aiheuttaa ongelmia. Mikäli palvelussa on erittäin paljon yhtäaikaisia käyttäjiä, palvelutaso voi ylikuormittua suosittelupyyntöjen suuresta määrästä. Palvelutaso voi myös kuormittua ja hidastua, mikäli datapisteiden määrä saavuttaa suuren datan rajan. (10.)

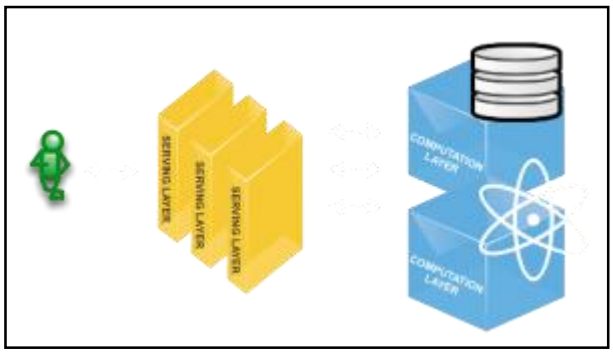


KUVA 4. Havainne itsenäisestä tilasta (6.)

3.2.2 Yhdistetty tila

Kun käytetään palvelu- ja laskentatasoa yhdessä, kutsutaan toimintatapaa yhdistetyksi tilaksi. Tällöin voidaan muodostaa useita palvelutasoa, jotka neuvottelevat useiden eri käyttäjien kanssa ja jotka välittävät pyyntöjä yhdelle tai use-

ammalle laskentatasolle. Tällöin pystytään hallitsemaan suurta dataliikennettä edestakaisin. (Kuva 5.)



KUVA 5. Havainne jaetusta tilasta (7.)

3.3 Myrrix-suosittelujärjestelmän käyttöönotto

Myrrix-suosittelujärjestelmän testaamista edeltävässä vaiheessa tuli ratkaista, täyttyvätkö suosittelujärjestelmän vaatimukset pelkällä itsenäisellä tilalla vai tarvitaanko suureen dataan kykenevää yhdistettyä tilaa. Oletuksena voitiin pitää, että itsenäisen tilan ajaminen kohdeympäristössä oli tarpeeksi tehokasta. Lisäksi päätettiin yhdistetyn tilan käyttöönotosta tilanteessa, jossa itsenäinen tila ei enää ole tarpeeksi tehokas. Näiden päätöksien pohjalta yhdistettyä tilaa ei tutkittu tarkemmin, vaan päätettiin keskittyä täysin itsenäisen tilan testaamiseen ja käyttöönottoon.

3.3.1 Itsenäisen tilan koeajo

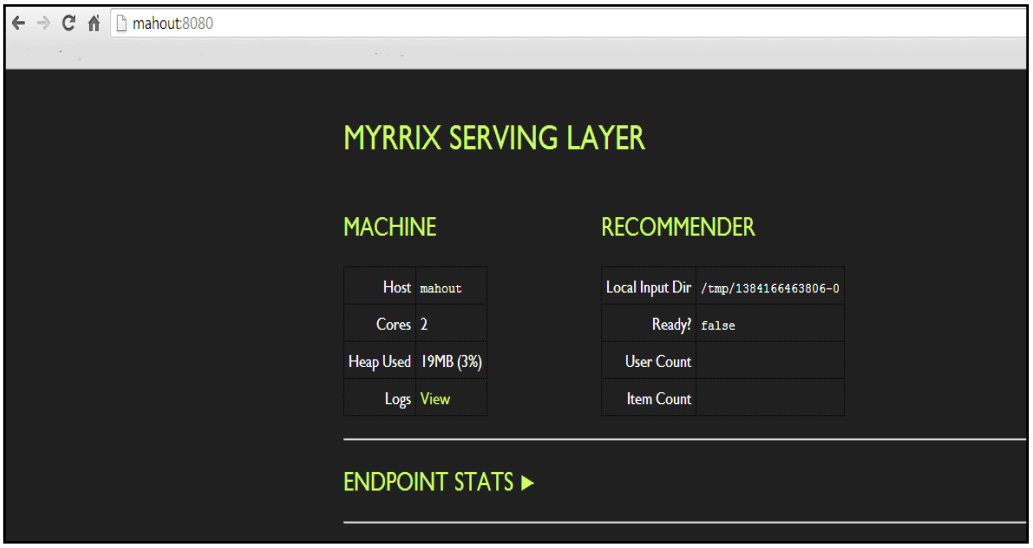
Myrrix-suosittelujärjestelmän itsenäisen tilan tutkiminen aloitettiin lataamalla suosittelujärjestelmä Myrrixin kotisivuilta. Suosittelujärjestelmän itsenäinen tila oli saatavissa JAR-pakettina, jota pystyttiin ajamaan millä tahansa koneella johon Java oli asennettu.

JAR-paketin lataamisen jälkeen se siirrettiin testiserverille joka oli varta vasten luotu suosittelujärjestelmän testaamista varten. Testiserverillä pystyttiin simuloimaan tilannetta, jossa Myrrix-suosittelujärjestelmää ajetaan tuotantopalvelimella. JAR-paketin ajaminen testiserverillä tapahtui yhdellä komentorivillä, joka ajettiin SSH-yhteyden kautta. (Kuva 6.)

```
java –Dmodel.features=100 –Dmodel.als.lambda=2 –Xmx512m –jar myrrix-serving-1.0.1.jar –port 8080
```

KUVA 6. Myrrix-suosittelujärjestelmän käynnistysparametrit

Komentorivin ajamisen jälkeen suosittelujärjestelmä oli käytännössä toimintavalmis. Parametrien avulla pystyttiin säätämään suosittelujärjestelmän toimintaa, mutta säätämiselle ei ollut tarvetta vielä tässä vaiheessa. Viimeisenä parametrina annettiin porttinumero, josta Myrrix-suosittelujärjestelmää pystyi kutsumaan. Palvelimen nimeä ja porttinumeroa käyttämällä pystyttiin yhdistämään Myrrix-suosittelujärjestelmän sisäiseen käyttöliittymään. (Kuva 7.)



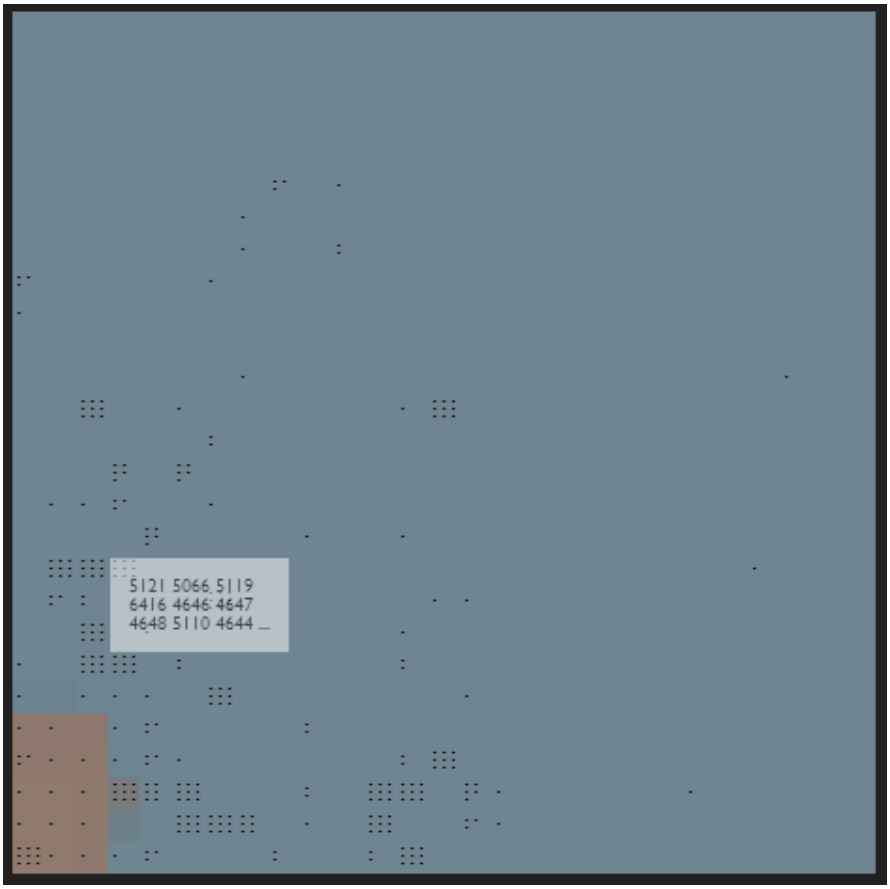
KUVA 7. Myrrix-suosittelujärjestelmän käyttöliittymä

Suosittelujärjestelmä ei pystynyt vastaamaan käyttäjän suosittelupyyntöihin vielä tässä vaiheessa, sillä järjestelmään ei ollut syötetty yhtään datapistettä. Testaamista varten käytettiin aiemmin kerättyä käyttäjätietoa, joka on nähtävissä kuvassa 2.

Myrrix-suosittelujärjestelmän graafinen käyttöliittymä on rakennettu erittäin helppokäyttöiseksi. Käyttöliittymän kautta voidaan testata erilaisia suosittelu-vaihtoehtoja ja sivustolle voidaan tulostaa yksittäisten käyttäjien suositeltavat

kappaleet. Lisäksi käyttöliittymästä löytyy muita toimintoja, joilla voidaan tulostaa aloitusdatan perusteella lasketut palvelun suosituimmat kappaleet.

Aloitusdatan syöttäminen käyttöliittymän kautta on erittäin nopea prosessi. Aloitusdatatiedoston pystyy selaamaan omalta koneeltaan suoraan järjestelmään, jonka jälkeen suosittelujärjestelmä prosessoi tiedoston. Prosessointiaika riippuu aloitusdatan koosta, mutta testikäytössä olleen datan prosessointiaika oli noin minuutti. Kuvassa 8 on havainnollistettu datamatriisia, johon datapisteet on sijoitettu algoritmin mukaisesti. Korostettuna on yksi tuoteryhmä, jonka sisältämät tuotetunnukset toimivat avaimina toisiinsa. Mikäli joku käyttäjä laulaa ryhmän sisältämän kappaleen, muut ryhmän kappaleet ovat laskennallisesti hyviä suosituksia käyttäjälle.



KUVA 8. Aloitusdatan datamatriisi

3.3.2 Myrrix-suosittelujärjestelmä ja PHP-kirjasto

Käytettävyyden kannalta oli olennaista ratkaista tapa, jolla suosittelujärjestelmään saataisiin luotettava ja helposti muodostettava yhteys, ja jota käyttäen karaokepalvelun käyttöliittymä kutsuisi suosittelujärjestelmää. Myrrixin kotisivuilta löytyi linkki kolmannen osapuolen tekemään PHP-kirjastoon, jonka avulla suosittelujärjestelmää pystyttiin kutsumaan PHP-skriptien avulla. (8.)

PHP-kirjaston avulla pystyttiin kirjoittamaan PHP-skriptejä, jotka pystyivät suoraan neuvottelemaan palvelutason kanssa ja välittämään tietoa sisään- ja ulospäin. PHP-kirjaston avulla Myrrix-suosittelujärjestelmän jokaista metodia, jotka oli esitelty rajapinnan sisällä, voitiin kutsua.

Kirjaston testaamista varten kirjoitettiin skripti, joka muodostaa rajapinnasta olion. Olion avulla voidaan käyttää kirjaston sisältä löytyviä metodeja, ja tulostaa niiden palauttamat arvot näkyviin. (Kuva 9.)

```
public function user_based_query($user_ID)

{

// Instanciate the Myrrix service

$myrrix = new MyrrixService('mahout', 8080);

// Get a recommendation for user

$recommendation = $myrrix->getRecommendation($user_ID); // an array of itemId and strength (example: [[325,0.53],[98,0.499]])

return json_encode($recommendation);

}
```

KUVA 9. PHP-rajapinnan testiskripti

Koska suosittelujärjestelmä palauttaa vain numeraalisia tietueita, palautusarvojen oikolukeminen oli työlästä. Oikolukua varten päätin kirjoittaa apuskriptin, joka hakee tietokannasta numeroarvoa vastaavan tietueen. (Kuva 10.)

```
$mysqli = new mysqli('matlock', 'jannen', '******', 'jannen');

if (mysqli_connect_error()) {

    die('Connect Error (' . mysqli_connect_errno() . ') '

    . mysqli_connect_error());

}

if (!$mysqli->set_charset("utf8")) {

    printf("Error loading character set utf8: %s\n", $mysqli->error);

} else {

}

$artist_name_query = "select song.id, song.title, artist.name from song inner
join artist on song.artist_id=artist.id where song.id in($mysql_clause)";

$fetching_artists_from_db_array = $mysqli>query($artist_name_query);

while($row = mysqli_fetch_assoc($fetching_artists_from_db_array))

{

$artists_list_from_database[] = $row["title"];

$songs_list_from_database[] = $row["name"];

$ids_list_from_database[]= $row["id"];

}



echo
json_encode(array("titlet"=>$artists_list_from_database,"namet"=>$songs_list_f
rom_database, "idt"=>$ids_list_from_database));
```

KUVA 10. Apuskripti tietokantoja varten

Skriptien testaamista varten tarvittiin HTML-sivusto, joka kutsui PHP-skriptejä AJAX-tekniikan avulla. HTML-sivustolle sijoitettiin tekstikenttä, jonka avulla kappaletunnukseen perustuvia suositteluja voitiin pyytää suosittelujärjestelmältä. Kappalepohjainen suosittelu vastaa tilannetta, jossa käyttäjä on järjestelmälle ennalta tuntematon ja hän on laulanut ensimmäisen kappaleensa.

Myrrix-suosittelujärjestelmään täytyi lisätä aloitusdataa ennen PHP-skriptien suorittamista. Aloitusdata oli suhteellisen pieni, noin 227 000 datapisteen suuruinen XLS-tiedosto. Aloitusdata oli kerätty palvelimen tietokannasta, johon tallennetaan jokainen kappale joka on käynnistetty palvelussa. Koska kantaan kirjoitetaan myös ne kappaleet joista kuunnellaan vain 30 sekunnin testiversio, ei aloitusdata ollut täysin luotettava. Lyhyet versiot olivat tarkoitettu uusien käyttäjien houkuttelemiseksi palveluun, ja ne aiheuttivat hajontaa suosittelujärjestelmässä.

Aloitusdata ei siis ollut optimaalinen, sillä sille ei tehty minkäänlaista oikolukua ennen palvelutasoon siirtämistä. Saatujen suosittelujen laatu oli silti erittäin korkeatasoinen, ja PHP-kirjaston toimivuus pystyttiin toteamaan. (Kuva 11.)

ITEM-ID Based recommendations 5110 //RecommendToAnonymous

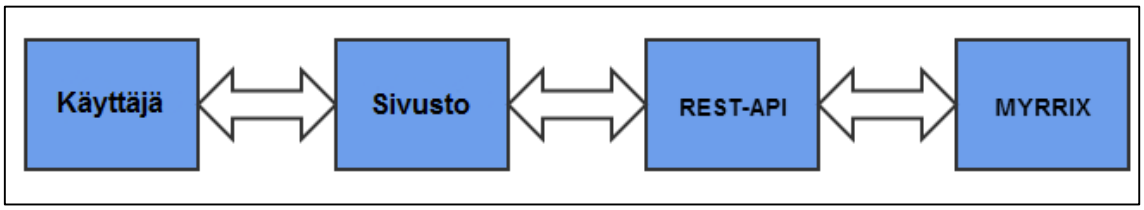
Submit

Artist Name	Track Name	SQL ID	Strength
30 Seconds To Mars	The Kill (Bury Me)	5009	0.03812158
Beyoncé	Listen	5033	0.02368597
Katy Perry	Teenage Dream	5102	0.02225478
Madonna	Papa Don't Preach	5125	0.02201734
Paramore	That's What You Get	5159	0.020893788
Backstreet Boys	As Long As You Love Me	6107	0.01809302
Bee Gees	Stayin' Alive	6108	0.01745797
Lady Gaga	Marry The Night	6151	0.017253853
Carly Rae Jepsen	Call Me Maybe	6596	0.01694278
Nicki Minaj	Starships	6617	0.016838547

KUVA 11. Suosittelut kappaleelle 5110

4 REST-RAJAPINTA JA MYRRIX-SUOSITTELUJÄRJESTELMÄ

Yrityksen kaikki taustaprosessit on integroitu yhteen yksittäiseen REST-tyyliseen rajapintaan. Sen avulla kaikki käyttäjän tarvitsemat funktiot hoidetaan REST-rajapinnan lävitse, eikä käyttäjä näe suoraan metodien sisällä olevia toimintoja. Rajapinnan käyttäminen on sekä tehokasta että turvallista, sillä käyttäjät eivät pääse käsiksi lähdekoodeihin eivätkä he pysty syöttämään haitallista ohjelmakoodia suoritusten väliin. (Kuva 12.)



KUVA 12. REST-rajapinnan sijoittuminen

4.1 Suosittelujärjestelmästä lähtevä liikenne

Suosittelujärjestelmän testaamisen jälkeen tuli laajentaa jo käytössä olevaa REST-rajapintaa siten, että se pystyy palvelemaan käyttäjiä myös suosittelujen muodossa. Tämä tehtiin kirjoittamalla rajapinnan sisään funktio, joka luo Myrrix-suosittelujärjestelmästä PHP-kirjaston avulla olion, jonka avulla voidaan käyttää suosittelujärjestelmän funktioita. (Kuva 13.)

```
static public function recommendations($uid) {
    $resource = new Resource(static::$valid_sort_by, AUTH::USER, static::$valid_fields);
    $myrrixport = Configuration::config('myrrix_port');
    $myrrixhost = Configuration::config('myrrix_host');
    $myrrix = new MyrrixService($myrrixhost, $myrrixport);
    $uid_int = (int)$uid;
    $recommendation = $myrrix->getRecommendation($uid_int);

    echo json_encode($recommendation);
  }
}
```

KUVA 13. REST-rajapinnan sisäinen funktio suosittelujärjestelmää varten

Rajapinnan testaamista varten luotiin PHP-skripti, joka otti yhteyttä rajapinnan funktioon. Tämä demonstroi tilannetta, jossa käyttäjän laukaisema funktio ottaa yhteyttä rajapintaan. Skriptin ajaminen ottaa yhteyden REST-rajapinnan verkko-osoitteeseen, josta suosittelujärjestelmäfunktio vastaa. Vastauksena saatiin käyttäjäkohtainen lista suosituksista, joten suosittelujärjestelmältä lähtevä liikenne rajapinnan välityksellä todettiin toimivaksi. (Kuva 14.)

```php
<?php
if (isset($_REQUEST['sendable_data']))
{
$uid = $_REQUEST['sendable_data'];
}
// create a new cURL resource
$ch = curl_init();
// set URL and other appropriate options
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_URL,
"http://mahout.greybrain.fi:8888/v1/index.php?__route__=/v1/users/$uid/recommendations");
curl_setopt($ch, CURLOPT_HTTPGET, true);
curl_setopt($ch, CURLOPT_HEADER, false);
curl_setopt($ch, CURLOPT_VERBOSE, false);
// grab URL and pass it to the browser
$recommendation = curl_exec($ch);
// close cURL resource, and free up system resources
curl_close($ch);
echo ($recommendation);
?>
```

KUVA 14. Suosittelujen noutoskripti REST-rajapinnan sisällä

4.2 Suosittelujärjestelmään tuleva liikenne

Seuraavaksi tuli luoda vastaavanlainen toiminnollisuus datan sisäänpäin syöttämistä varten. REST-rajapinnan sisälle tuli kirjoittaa toinen funktio, jossa käyttäjätunnus ja lauletun kappaleen tunnus lähetettiin suosittelujärjestelmälle. Lisäksi tulevia jälkiprosessointivaiheita ennakoiden lähetettiin kolmas arvo, joka toimi datapisteen painoarvona. (Kuva 15.)

Funktion testaaminen toteutettiin samalla periaatteella kuin edellisen funktion. Datan perillemeno vahvistettiin Myrrix-suosittelujärjestelmän oman käyttöliittymän kautta, ja pystyttiin varmistumaan siitä, että REST-rajapinta oli valmis hyödyntämään suosittelujärjestelmän toimintoja.

```
static public function myrrix_setpreference($uid, $sid) {
    $myrrixport = Configuration::config('myrrixport');
    $myrrixhost = Configuration::config('myrrixhost');
    $myrrix = new MyrrixService($myrrixhost, $myrrixport);


    try
    {
    $myrrix->setPreference($uid, $sid);
    }
    catch(Exception $e)
    {
    return false;
    }
```

KUVA 15. Datapisteiden lähetysskripti REST-rajapinnan sisällä

5 MYRRIX-SUOSITTELUJÄRJESTELMÄN OPTIMOINTI

Edellisten vaiheiden suorittamisen jälkeen suosittelujärjestelmän rakentaminen oli edennyt siihen vaiheeseen, että perustoiminnollisuus oli kunnossa. Suosittelujärjestelmä kykeni toimimaan REST-rajapinnan välityksellä, ja dataa käytettiin käyttöliittymän puolella antamaan käyttäjille suositteluja.

Suosittelujen laadun takaamiseksi täytyi luoda toiminnallisuus suosittelujen jälkiprosessointia varten. Käyttäjä saa pyytäessään 10 kappaletta suosittelujärjestelmältä, mutta järjestelmä ei osannut tässä vaiheessa erottaa, onko käyttäjä laulanut suositeltuja kappaleita vai ei. Tavoitteena oli luoda logiikka, jonka avulla prosessoidaan suosittelujärjestelmältä tulevaa suosittelulistaa, ja näytetään käyttäjälle vain ne, joita hän ei ole viime aikana laulanut.

5.1 Redis-järjestelmän käyttöönotto

Redis-järjestelmä on avoimeen lähdekoodiin perustuva tietokanta, jota ajetaan palvelimella. Redis toimii kuten vastaava relaatiotietokanta, kuten mm. MySQL, mutta sitä suoritetaan suoraan muistista. (9.)

Redis-järjestelmän implementointitarve tuli käyttäjäkohtaisen datan säilyttämisestä. Kun käyttäjä laulaa kappaleen, tallennetaan se Redis-järjestelmään. Redis-järjestelmän listassa pystytään kierrättämään 10 arvoa, jonka täyttyessä uusin syötetty arvo poistaa vanhimman. Tämän avulla pystyttiin tallentamaan käyttäjän viimeiset 10 laulamaa kappaletta, ja kyettiin sen avulla jälkiprosessoimaan suosittelulistaa.

Redis-järjestelmää testattiin SSH-yhteyden välityksellä testipalvelimella. Palvelimella käynnistettiin Python-ohjelmointikielen kääntäjä, jossa Redis-järjestelmä otettiin käyttöön ja alustettiin. (Kuva 16.)

```
Last login: Fri Nov 22 11:38:45 2013 from jannen-hp.greybrain.fi
jannen@mahout:~$ python
Python 2.7.3 (default, Sep 26 2013, 16:35:25)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>> r=redis.StrictRedis(host='localhost', port=6379)
>>> r.set('a', 1)
True
>>> r.get('a')
'1'
```

KUVA 66. Redis-järjestelmän testaaminen

Datan tallentaminen REST-rajapinnan välityksellä Redis-järjestelmään tapahtui PHP-skriptillä lähes samalla tavalla (kuva 17).

```
static public function redis_setvalue($uid, $sid) {
$redis = new Predis\Client(Configuration::config('redis'));
$redis->lpush($uid, $sid);
}
```

KUVA 17. Redis-järjestelmän päivittäminen PHP-skriptin kautta

REST-rajapinnan tuki Redis-järjestelmän datansyöttöä varten oli edellä mainittujen vaiheiden suorittamisen jälkeen valmis. Kun rajapinta päivittää Myrrix-suosittelujärjestelmää käyttäjän laulamalla kappaleella, rajapinta välittää myös kappaleen tunnuksen Redis-järjestelmään.

Datan nouto Redis-järjestelmästä toteutettiin saman funktion sisällä, jossa Myrrix-suosittelujärjestelmältä noudetaan suosittelulista käyttäjäkohtaisesti. Kun suosittelujärjestelmä palauttaa käyttäjäkohtaisen suosittelulistan, noudetaan Redis-järjestelmältä lista viimeksi lauletuista kappaleista. Näiden kahden listan vertailulla voidaan poistaa suositteluista ne kappaleet, joita käyttäjä on laulanut viimeisen 10 kappaleen aikana. (Kuva 18.)

```
static public function cleanRecommendations($recommendations, $uid)
  {
  $redis = SORedis::getRedis();
  $userID = "api:recommendation:$uid";
  $recentlyplayed = $redis->lrange($userID, 0, 9);

  $songids = array_map(function($val) {return $val[0]; }, $recommendations);

  $songids = array_filter($songids,
   function($val) use ($recentlyplayed) { return !in_array($val, $recentlyplayed); });

   $songids = Songs::areValidSongIds($songids);
   $recommendations = array_filter($recommendations,
     function($val) use ($songids) {return in_array($val[0], $songids); } );
   return array_slice($recommendations, 0, 10);
  }
```

KUVA 78. Suosittelujen jälkiprosessointia Redis-järjestelmän avulla

5.2 Suosittelujen määrän optimointi

Suosittelujärjestelmä palautti oletusarvoisesti 10 suosittelua käyttäjälleen, ja Redis-järjestelmässä tallennettiin 10 edellistä laulettua kappaletta. Yhdeksi käyttäjäongelmaksi olisi voinut muodostua tilanne, jossa käyttäjä on laulanut juuri ne 10 kappaletta, jotka suosittelujärjestelmä käyttäjälle suosittelisi. Tilanteen muokkaamiseksi täytyi laajentaa PHP-kirjastoa, ja suurentaa suosittelujenmääräparametria yrityksen tilanteeseen soveltuvaksi. (Kuva 19.)

```
public function getExtendedRecommendation($userId)
{
$command = $this->client->getCommand('GetRecommendation', array(
      'userID'  => $userId,
      'considerKnownItems'  => 'true',
      'howMany' => 20
   ));
return $this->client->execute($command)->json();
}
```

KUVA 19. PHP-kirjaston laajennusosa

Laajennusosan avulla Myrrix-suosittelujärjestelmä palautti 10 suosittelun sijaan 20 suositeltavaa kappaletta. Nyt tilanne, jossa käyttäjä olisi laulanut 10 suosittaluksi luokiteltavaa kappaletta, ei aiheuta REST-rajapinnan kaatumista, vaan tulostaa silti 10 jäljelle jäänyttä suositeltavaa kappaletta käyttäjälle.

5.3 Suosittelujen painoarvon käyttöönotto

Suosittelujärjestelmän kolmatta parametria eli painoarvoa ei vielä käytetty suosittelujen parantamiseen. Oletusarvoisesti Myrrix-suosittelujärjestelmä asettaa jokaiselle datapisteelle painoarvoksi 1.0, josta arvo voi heitellä vapaasti valittavan vaihteluvälin sisällä. Painoarvon avulla suosittelujärjestelmä lajittelee suosittelunsa paremmuusjärjestykseen.

Jotta käyttäjän laulaman kappaleen painottaminen tapahtuisi oikein, tuli suunnitella logiikka, joka painottaisi käyttäjän laulamaa kappaletta sitä paremmin, mitä pidemmälle käyttäjä laulua laulaa. Täten heti alkuvaiheessa lopetettu kappale antaisi käyttäjälleen erittäin huonon painoarvon, ja loppuun asti laulettu kappale erittäin hyvän painoarvon.

REST-rajapinta seurasi jo ennestään käyttäjän laulamaa kappaletta ja antoi palautusarvoksi myös prosenttilukeman suoritetusta kappaleesta. Prosenttilukemaa käytettiin hyväksi painoarvon luomisessa. Painoarvon raja-arvoiksi valittiin väli 1–4. Datapisteiden lähetysskriptiä muokkaamalla pystyttiin välittämään käyttäjä- ja tuotetunnuksen lisäksi myös painoarvo. (Kuva 20.)

```
static public function myrrix_setpreference($uid, $sid, $percentage) {
    $myrrixport = Configuration::config('myrrixport');
    $myrrixhost = Configuration::config('myrrixhost');
    $myrrix = new MyrrixService($myrrixhost, $myrrixport);

    $percentage = (float) ($percentage / 100) * 4;
    try
    {
    $myrrix->setPreference($uid, $sid, $percentage);
    }
    catch(Exception $e)
    {
    return false;
    }
```

KUVA 208. Prosenttilukeman muunto painoarvoksi REST-rajapinnan sisällä

6 SUOSITTELUJÄRJESTELMÄN VIIMEISTELY

6.1 REST-rajapinnan refaktorointi

Suosittelujärjestelmän eri toiminnollisuuksien implementoinnin jälkeisessä vaiheessa keskityttiin kokonaisen järjestelmän viimeistelyyn ja testaamiseen. Testaamisessa koeajettiin REST-rajapintaa, jossa aikaisemmin luodut funktiot toimivat.

Suurinta jatkokehitystä tapahtui REST-rajapinnan sisällä. Rajapinnan laajentaminen vaiheittain aiheutti sen, ettei funktioiden rakenne ollut hyvin ryhmiteltynä. Rajapinnalle suoritettiin kattava refaktorointi, jossa toiminnallisuus säilyi ennallaan, mutta sisäinen rakenne parani huomattavasti. Liitteessä 1 on nähtävissä opinnäytetyön aikana kirjoitettu REST-rajapintalaajennus, johon on keskitetty kaikki suosittelujärjestelmään liittyneet funktiot.

6.2 Myrrix-suosittelujärjestelmän PHP-rajapinnan refaktorointi

Myrrix-suosittelujärjestelmän PHP-rajapinnan toiminnallisuutta tuli muokata REST-rajapinnan refaktoroinnin muuttaessa funktioiden kutsumistapaa. Hyvän ohjelmointiperiaatteen ja helpon ylläpidettävyyden vuoksi laajensin suosittelujärjestelmän PHP-rajapintaa omalla skriptillä, jossa omaan käyttötarkoitukseen muokatut funktiot toimivat. Kun Myrrix-suosittelujärjestelmän PHP-rajapintaa päivitetään, ei tarvitse huolehtia funktioiden uudelleenkirjoittamisesta jokaisen päivityksen yhteydessä erikseen, vaan voi hyödyntää PHP-rajapinnalle kirjoitettua laajennusta, joka säilyy entisellään päivityksistä huolimatta. Liitteessä 2 on nähtävissä laajennusosa PHP-rajapinnalle.

7 POHDINTA

Opinnäytetyön tavoitteena oli toteuttaa SingOn-yrityksen palveluun suosittelujärjestelmä, joka suosittelisi käyttäjilleen kappaleita laulettavaksi. Suosittelujärjestelmän tulisi myös olla kyvykäs oppimaan uutta dataa käyttäjiltä ja näin päivittää itse itseään, jotta suosittelut pysyisivät tuoreina.

Työn alkuvaiheen tutkimusvaiheessa tulikin ilmi laaja suosittelujärjestelmien kirjo, ja valinnan tekemistä vaikeutti se, ettei yksikään järjestelmä ollut selvästi muita parempi. Valinta tehtiinkin perusteellisen tutkimisen ja käytettävissä olevan aikataulun mukaisesti. Implementaatioon ja mahdolliseen modifiointiin varattu aika oli vain arvio, ja todellista ajantarvetta ei pystytty luotettavasti laskemaan.

Varsinaiseen tutkimusvaiheeseen tarvittavaa materiaalia löytyi internetistä erittäin kattavasti. Voinkin olettaa, että näin jatkuvasti kehittyvässä alueessa (ja alalla) kirjallinen materiaali jää hyvin nopeasti vanhaksi, eikä kirjoista saa perusidean lisäksi muuta hyödynnettävää materiaalia.

Kun valitsin Myrrix-suosittelujärjestelmän, löysin erittäin kattavan sivuston järjestelmän implementoinnista. Suosittelujärjestelmän kotisivuilla oli kattava dokumentaatio, ja musiikkiaiheiset esimerkit saattoivatkin olla merkittävässä roolissa kun suoritin lopullista valintaa.

Opinnäytetyön aikana Myrrix-suosittelujärjestelmän kehittäjät fuusioituivat suuremman yrityksen kanssa, ja Myrrix-suosittelujärjestelmän kehittäminen sellaisenaan loppui. Myrrix-suosittelujärjestelmän elinkaari jatkunee silti useamman vuoden eteenpäin, mutta fuusioituneen yrityksen tarjonnasta löytyy Myrrix-suosittelujärjestelmän seuraaja, Oryx. Vaikka jatkokehitystä voitaisiin tehdä Oryxin suuntaan, ei tässä vaiheessa tehty vielä päätöstä.

LÄHTEET

1. Ho, Ricky 2011. Pragmatic Programming Techniques - Recommendation Engine. Saatavissa: http://horicky.blogspot.fi/2011/09/recommendation-engine.html. Hakupäivä 20.11.2013

2. Ullman, Jeffrey 2013. Chapter 9. Recommendation Systems. Teoksessa - Mining of Massive Datasets. Stanford University. Saatavissa: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf. Hakupäivä 21.11.2013

3. Oracle Corp. 2002. Figure 5-6. Saatavissa: http://isu.ifmo.ru/docs/IAS904/core.904/b10375/img/per_arch_bi.gif. Hakupäivä 21.11.2013

4. Lineaarinen regressioanalyysi. 2012. Wikipedia. Saatavissa: http://upload.wikimedia.org/wikipedia/commons/thumb/b/be/Normdist_regression.png/300px-Normdist_regression.png. Hakupäivä 23.12.2013

5. Myrrix. 2013. Myrrix Corp. Saatavissa: http://myrrix.com/. Hakupäivä 22.11.2013

6. Stand-alone mode. 2013. Myrrix Corp. Saatavissa: http://myrrix.com/wp-content/uploads/2012/03/Serving-300x258.png. Hakupäivä 22.11.2013

7. Distributed mode. 2013. Myrrix Corp. Saatavissa: http://myrrix.com/wp-content/uploads/2012/03/Distributed-300x169.png. Hakupäivä 22.11.2013

8. Salib, Michael. 2013. Myrrix PHP-client. Github. Saatavissa: https://github.com/michelsalib/bcc-myrrix. Hakupäivä 25.11.2013

9. Redis. 2013. Saatavissa: http://redis.io/ Hakupäivä 2.12.2013

10. Example: Performance. 2013. Myrrix Corp. Saatavissa: http://myrrix.com/example-performance/. Hakupäivä 1.2.2014

```
<?php

use BCC\Myrrix\MyrrixService;
use BCC\Myrrix\MyrrixExtension;

class Recommendations {

protected static $cache = null;

static private function cache() {
        if(!self::$cache) {
            self::$cache = new Memcached();
            if(!self::$cache->addServer(Configuration::config('memcache_host'),
Configuration::config('memcache_port')))
                error_log("Memcached addServer failed");
        }
        return self::$cache;
    }

static public function myrrix_setpreference($uid, $sid, $percentage) {
        $myrrixport = Configuration::config('myrrixport');
        $myrrixhost = Configuration::config('myrrixhost');
        $myrrix = new MyrrixService($myrrixhost, $myrrixport);

        $percentage = (float) ($percentage / 100) * 4;
        try
        {
        $myrrix->setPreference($uid, $sid, $percentage);
        }
        catch(Exception $e)
        {
        return false;
        }


        }

static public function redis_setpreference($uid, $sid)
    {
    $redis = SORedis::getRedis();
    $pipe = $redis->pipeline();

    $userID = "api:recommendation:$uid";

    $pipe->lpush($userID, $sid);
    $pipe->ltrim($userID,0,9);
    $pipe->execute();
    }

static public function myrrix_getrecommendations($uid)
```

```
    {
    $myrrixport = Configuration::config('myrrixport');
    $myrrixhost = Configuration::config('myrrixhost');
    $myrrix = new MyrrixExtension($myrrixhost, $myrrixport);

    try
        {
        $recommendations = $myrrix->getExtendedRecommendation(intval($uid));
        return self::cleanRecommendations($recommendations, $uid);
        }
    catch (\Guzzle\Http\Exception\ClientErrorResponseException $e)
        {
        $recommendations = self::myrrix_getToplist();
        return self::cleanRecommendations($recommendations, $uid);
        }
    catch (Exception $e)
        {
        return Array();
        }

    }

static private function myrrix_getToplist()
    {

    $recommendations = self::cache()->get('MyrrixTopList');
    if(!$recommendations)
        {
        $myrrixport = Configuration::config('myrrixport');
        $myrrixhost = Configuration::config('myrrixhost');
        $myrrix = new MyrrixExtension($myrrixhost, $myrrixport);
        try
            {
            $recommendations = $myrrix->getMostPopularItems();
            self::cache()->set('MyrrixTopList', $recommendations);
            return $recommendations;
            }
        catch(Exception $e)
            {
            return Array();
            }
        }
    else
        {
        return $recommendations;
        }
    }

static public function cleanRecommendations($recommendations, $uid)
    {
```

```
    $redis = SORedis::getRedis();
    $userID = "api:recommendation:$uid";
    $recentlyplayed = $redis->lrange($userID, 0, 9);

    $songids = array_map(function($val) {return $val[0]; }, $recommendations);

    $songids = array_filter($songids,
     function($val) use ($recentlyplayed) { return !in_array($val, $recentlyplayed);
});

     $songids = Songs::areValidSongIds($songids);
     $recommendations = array_filter($recommendations,
        function($val) use ($songids) {return in_array($val[0], $songids); } );
      return array_slice($recommendations, 0, 10);
    }
}
?>
```

```
<?php

namespace BCC\Myrrix;

class MyrrixExtension extends MyrrixService
{

   function __construct($host, $port, $username = null, $passwd = null)
   {
                              parent::__construct($host, $port, $username,
$passwd);
   }

//@param $uid int - User ID
//@param $consider string - "true" for considering known items -> Hardcod-
ed to 'true'
//@param $count int - how many recommendations are returned -> Hard-
coded to 20.

   public function getExtendedRecommendation($userId)
   {
   $command = $this->client->getCommand('GetRecommendation', array(
         'userID'  => $userId,
         'considerKnownItems'  => 'true',
         'howMany' => 20
      ));
   return $this->client->execute($command)->json();
   }


   public function getMostPopularItems()
   {
   $command = $this->client->getCommand('GetMostPopularItems', array(
         'howMany' => 20
         ));
   return $this->client->execute($command)->json();

   }
}
?>
```